Bloque II: El nivel de aplicación

Tema 3: Protocolos del nivel de aplicación I

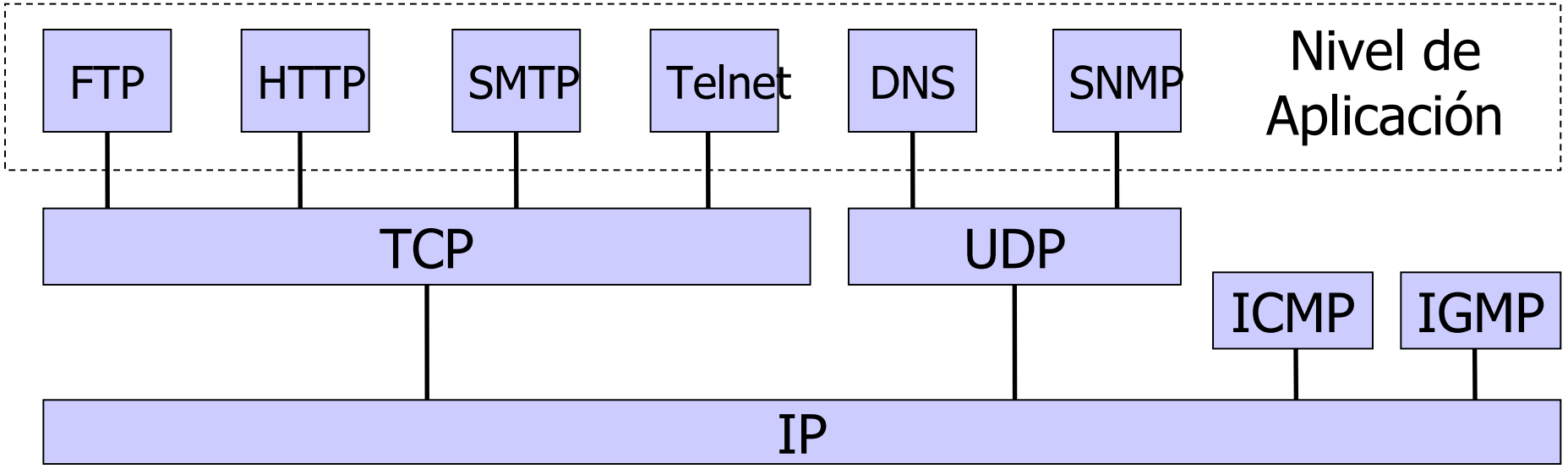
Índice

- Bloque II: El nivel de aplicación
  - Tema 3: Protocolos del nivel de aplicación I
    - Introducción
    - World Wide Web
    - Correo electrónico

- **Lecturas recomendadas:**
  - Capítulo 2, secciones 2.2 y 2.4, de “Redes de Computadores: Un enfoque descendente”. James F. Kurose, Keith W. Ross. Addison Wesley.

Introducción

- Dos procesos en dos sistemas finales (distintos) se comunican intercambiando mensajes a través de una red de computadores.
- Modelo **cliente-servidor**:
  - Cliente envía peticiones al servidor (solicitando un servicio).
  - Servidor recibe las peticiones, las procesa y envía la respuesta.
- Modelo **Peer to peer**: los dos extremos realizan un servicio y solicitan servicios.
- Protocolos del nivel de aplicación:
  - Definen el formato y el orden de intercambio de los mensajes
  - Acciones en la transmisión o recepción de mensajes



Web

- Es una aplicación/servicio más de Internet.
- Combinaba cuatro ideas que no eran nuevas:
  - **Hipertexto**: formato de la información que permite moverse de una parte a otra de un documento o entre documentos mediante conexiones internas entre estos documentos (**hiperenlaces** o **enlaces**).
  - **Identificadores de recursos**: identificadores únicos que permiten localizar un recurso en la red (URL – Uniform Resource Locator o URI – Uniform Resource Identifier)
  - **Modelo cliente-servidor**
  - **Lenguaje de marcas**: caracteres o códigos embebidos en texto que indican estructura, semántica o recomendaciones para su presentación (HTML – HyperText Markup Language).

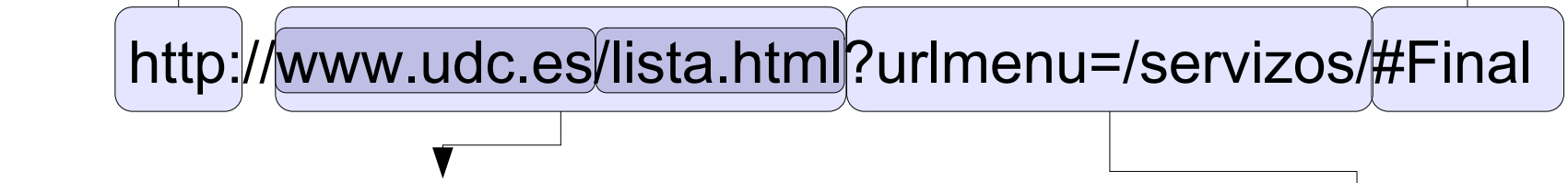
- Componentes:
  - **Página Web**: archivo HTML base + objetos (p.e. imágenes)
  - **Navegador**: agente de usuario para el Web
  - **Servidor Web**: almacena objetos Web direccionables a través de una URL
  - **Protocolo HTTP**: permite comunicarse al servidor y al navegador

URI

- URI: identificador que permiten acceder a un recurso web, por ejemplo, una página web.
- Estructura:

Esquema: especifica el protocolo utilizado para acceder al recurso. Por ejemplo, http, ftp, https, mailto, ...

Fragmento (opcional): identifica una subdirección dentro de un recurso.



Parte jerárquica:

- **Autoridad**: nombre (o dirección IP) del servidor. Puede incluir el número de puerto o información de control de acceso.
- **Ruta**: para acceder al recurso. Similar a los directorios.

Consulta (opcional): información adicional, normalmente variables y sus valores. Por ejemplo, para enviar los campos de un formulario.

URL vs. URI

- Normalmente denominamos URLs a todas las direcciones de recursos en Internet.
- Aunque el URI es más completo que la URL:
  - URL = URI – Fragmento

- Ejemplos:
  - http://www.fic.udc.es
  - http://www.fic.udc.es/
  - http://www.fic.udc.es:80/
  - https://www.fic.udc.es/gl/presentacion
  - https://www.udc.es/gl/sobreUDC/
  - mailto:john.doe@udc.es

HTTP

- HyperText Transfer Protocol
- Especificado en RFC 1945 (HTTP/1.0), RFC 2616 (HTTP/1.1), RFC 7540 (HTTP/2) y RFC 9114 (HTTP/3)
  - Compatible con versiones anteriores
  - Aunque HTTP/2 no es compatible en la transmisión
- Utiliza el protocolo TCP (servicio orientado a conexión y fiable) → Cada mensajes HTTP emitido por el cliente o servidor llega al otro extremo sin modificaciones.
- Define cómo los clientes (navegadores) solicitan páginas Web y cómo los servidores transfieren estas páginas:
  - El cliente envía una **petición** HTTP
  - El servidor contesta con **respuesta** HTTP
- HTTP es un protocolo sin estado → El servidor no almacena información sobre las peticiones anteriores del cliente.

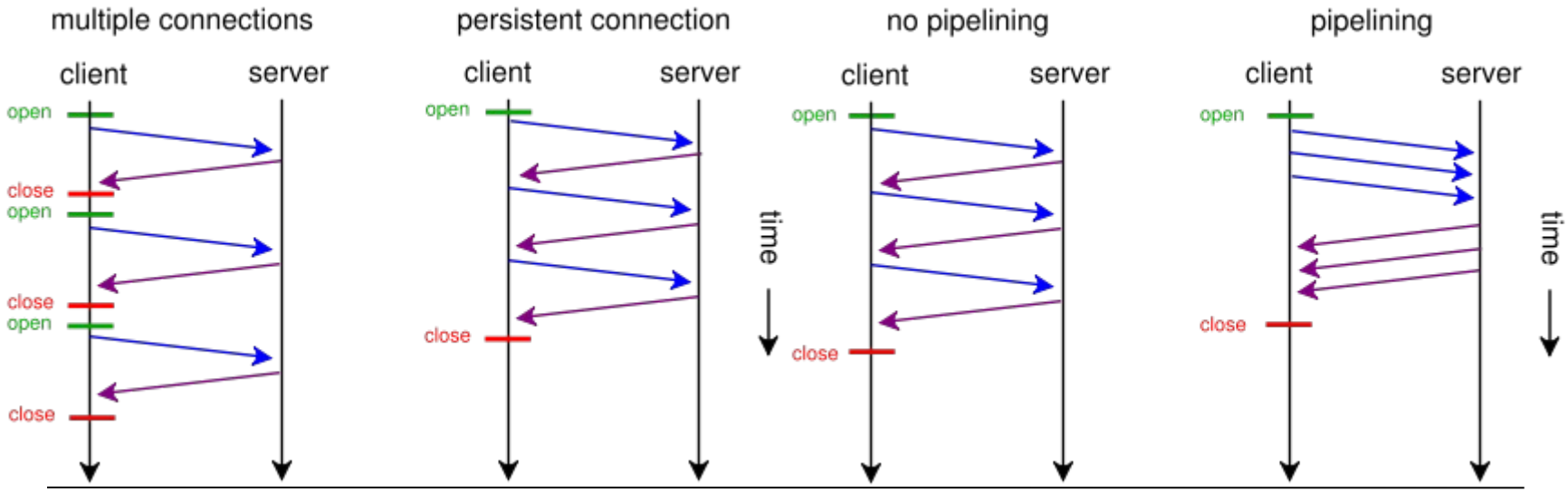
HTTP: Conexiones no persistentes

- HTTP/1.0 usa conexiones no persistentes
- Petición de una URL (http://www.fic.udc.es/presentacion)
  1. El cliente HTTP inicia la conexión TCP con el servidor www.fic.udc.es en el puerto 80.
  2. El cliente HTTP envía al servidor el mensaje de petición solicitando el objeto **/presentacion** (el fichero por defecto que se encuentra en /presentacion).
  3. El servidor HTTP recibe la petición, busca el objeto, lo encapsula en el mensaje HTTP de respuesta y lo envía.
  4. El servidor finaliza la conexión TCP.
  5. El cliente HTTP recibe la respuesta y finaliza la conexión TCP.
  6. El cliente extrae el archivo del mensaje de respuesta, examina el archivo HTML y encuentra referencias a otros objetos HTML (p.e. imágenes)
  7. Para cada objeto, volver al paso 1.

- Dependiendo del navegador, las nuevas conexiones podrían ser en paralelo.
- Inconvenientes:
  - Se necesita una conexión (buffers, variables, timeouts, ...) para cada objeto solicitado.
  - Retardo de dos veces el RTT (Round-Trip Time): establecimiento de conexión + petición y recepción del objeto

HTTP: Conexiones persistentes

- Por defecto, en HTTP/1.1
- El servidor HTTP deja abierta la conexión TCP, esperando nuevas petición/respuestas.
  - El servidor cerrará la conexión después de un tiempo de inactividad.
- Sin pipeline: el cliente sólo envía una nueva petición cuando ha recibido la respuesta previa.
- Con pipeline: el cliente realiza una petición tan pronto encuentra una referencia a un objeto.

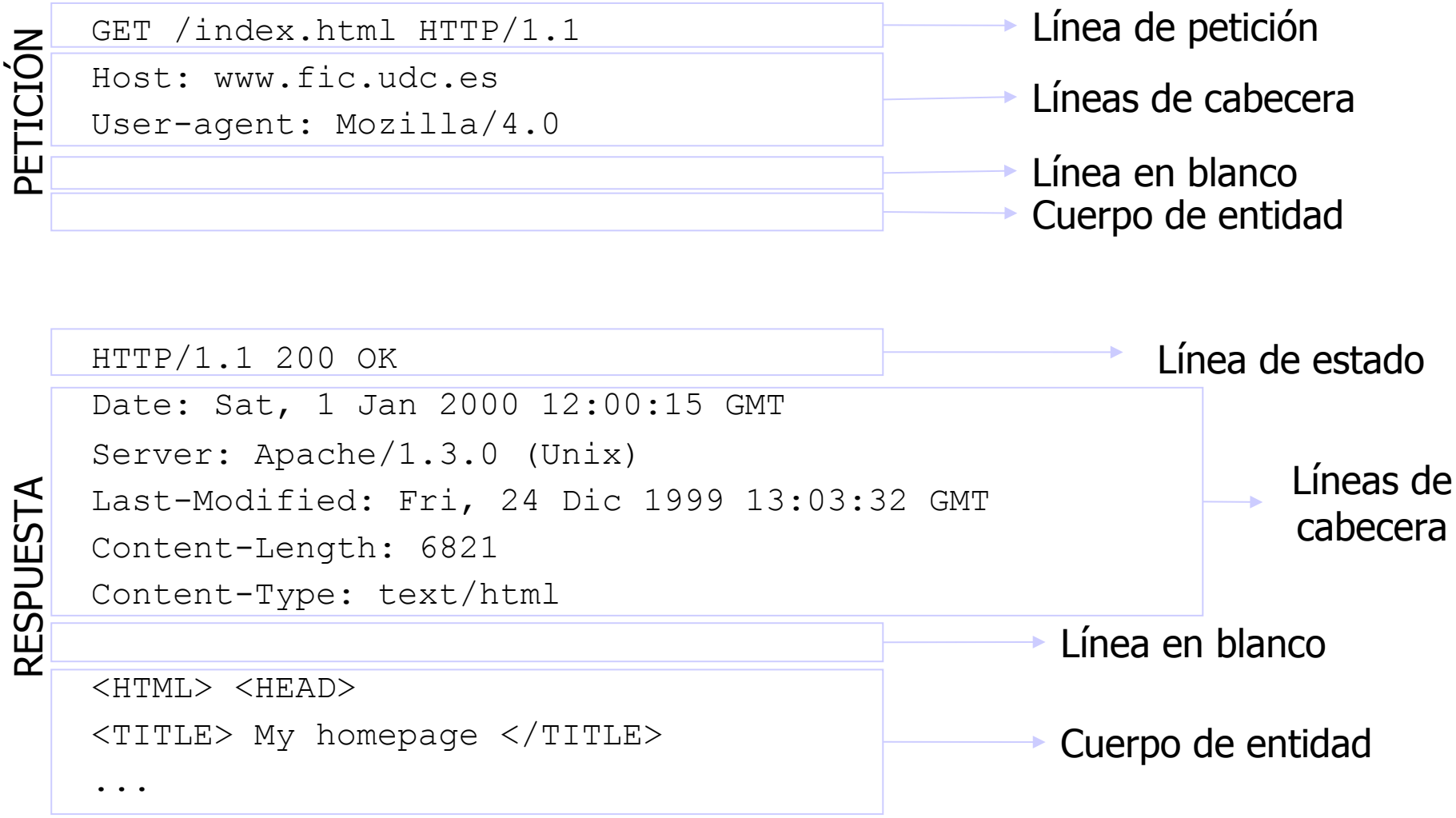


HTTP/2 y 3

- **HTTP/2** – RFC 7540 (mayo 2015)
  - Se basa en el protocolo **SPDY** de Google
  - No cambia el protocolo: métodos, códigos de estado, … son los mismos. Cambia la manera en la que se envían los datos.
  - Mejoras:
    - **Multiplexación total** sobre una conexión TCP: descarga de objetos web asíncronamente. Soluciona problema head-of-line (HOL) en HTTP/1.1.
    - Protocolo en **formato binario** y **compresión de cabeceras**.
    - **Server Push**: el servidor puede enviar objetos no solicitados por el cliente para almacenar en caché.

- **HTTP/3** – RFC 9114 (junio 2022)
  - Utiliza el protocolo **QUIC** de Google → QUIC (Quick UDP Internet Connections): un protocolo de código abierto basado en UDP.
  - Mejoras: mantiene multiplexaxión total, **mejor latencia**, **control de flujo por stream** e incluye Transport Layer Security (TLS) 1.3.

Mensajes HTTP



Petición HTTP

- Línea de petición + línea en blanco: obligatorio
- Línea de petición: Método URL HTTP/Versión
  - Método:
    - GET: utilizando cuando el navegador solicita un objeto.
    - HEAD: el servidor responde con un mensaje HTTP, pero sin incluir el objeto solicitado (sólo la cabecera).
    - POST: incluye datos en el cuerpo de entidad (frente al GET que los codifica en la URL).
    - PUT (1.1): permite a un usuario cargar un objeto en la ruta especificada.
    - DELETE (1.1): permite borrar un objeto de un servidor Web.
  - URL: objeto al que se hace referencia
  - Versión
- Host: especifica el host en el que reside el objeto.
- User-agent: especifica el tipo de navegador que está haciendo la petición.

- POST: utilizado comúnmente cuando un usuario rellena un formulario.
  - El cuerpo de entidad contiene los datos introducidos por el usuario.
- GET: también soporta el envío de datos introducidos por el usuario.
  - Se envían codificados en la URL real.
  - Por ejemplo: www.google.com/search?keywords=information+retrieval

Respuesta HTTP

- **Línea de estado**: Versión + Código de estado + Frase
- Códigos de estado agrupados en 5 tipos:
  - Informativo: 1xx. P.e. 100 Continue
  - Éxito: 2xx. P.e. 200 OK
  - Redirecciones: 3xx. P.e. 301 Moved Permanently
  - Error del cliente: 4xx. P.e. 404 Not Found
  - Error del servidor: 5xx. P.e. 500 Internal Server Error
- https://en.wikipedia.org/wiki/List_of_HTTP_status_codes

Líneas de cabecera

- Date: fecha y hora en la que se creó y envió la respuesta HTTP.
- Server: especifica el tipo de servidor Web que ha atendido a la petición.
- Last-Modified: indica la fecha y hora en que el objeto fue creado o modificado por última vez.
- Content-Length: indica el número de bytes del objeto enviado.
- Content-Type: indica el tipo de objeto incluido en el cuerpo de entidad.
  - La extensión del archivo no especifica (formalmente) el tipo de objeto.
- https://en.wikipedia.org/wiki/List_of_HTTP_header_fields

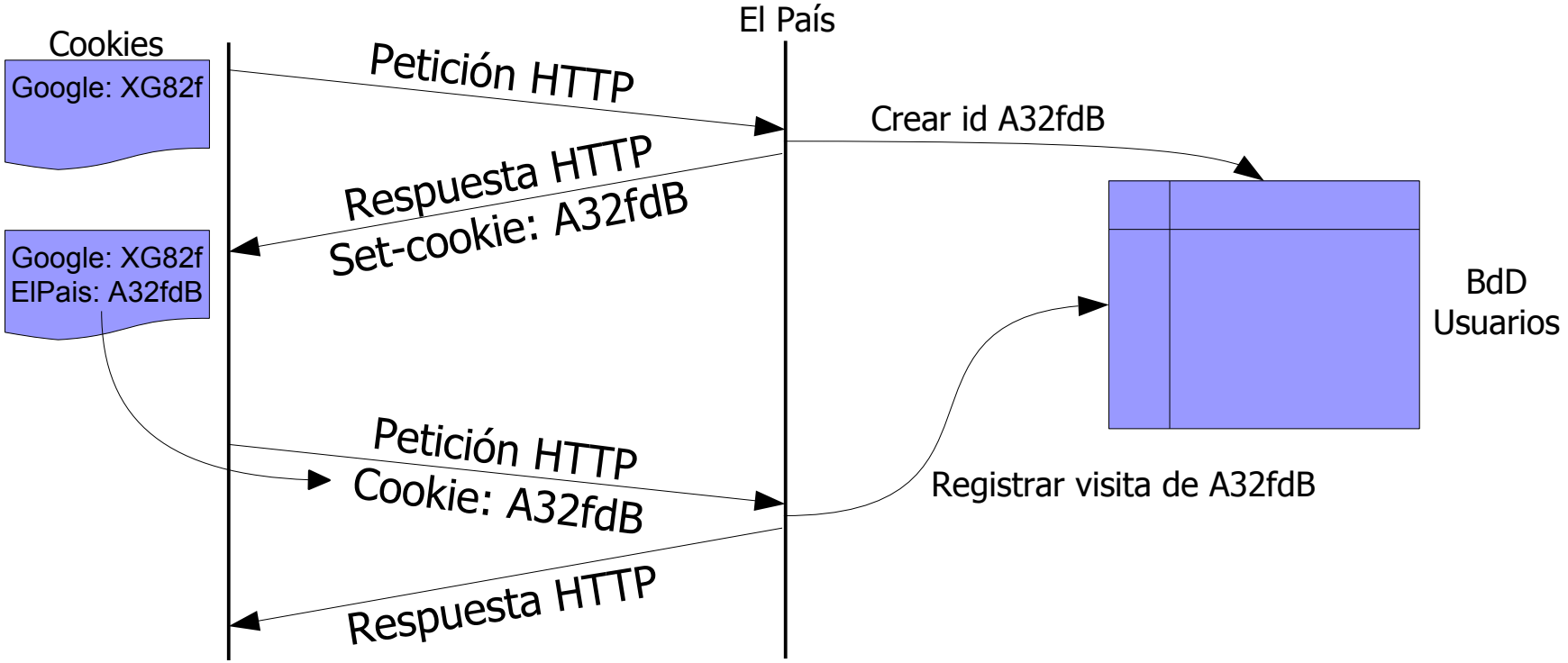
Respuesta HTTP: Content-Type

- Usa los tipos MIME (Multipurpose Internet Mail Extensions) para definir el tipo de contenido.
- **MIME**: estándar que indica el tipo del contenido. Gestionado por la IANA:
  - https://www.iana.org/assignments/media-types/media-types.xhtml
- Estructura básica: tipo/subtipo (p.e. text/html, image/gif, ...)

- **Discreto**: un único documento de un único tipo
  - application: application/pdf, application/zip, application/octet-stream
  - audio: audio/mpeg
  - image: image/jpeg, image/png, image/gif
  - text: text/plain, text/html, text/csv
  - video: video/mp4
- **Multipart**: encapsula múltiples archivos (posiblemente de distintos tipos) en una única transacción.

HTTP: Cookies

- HTTP no tiene memoria → Entonces, ¿cómo se "acuerdan" de mí los servidores web?
- Cookies (RFC 2965): mecanismo que permite a un servidor web guardar información en mi navegador.



HTTP: GET condicional

- La utilización de una caché reduce los retardos de recuperación de objetos y reduce el tráfico que circula por la red.
- Problema: la copia de un objeto en caché puede ser obsoleta.
- Solución: GET + If-Modificed-Since
  - Sólo devuelve el objeto si ha sido modificado después de la fecha indicada.

- Solicitar un objeto por primera vez:

```
GET /images/udc.gif HTTP/1.1
User-agent: Mozilla/4.0
```

- Recibir la respuesta del servidor:

```
HTTP/1.1 200 OK
Date: Sat, 1 Jan 2000 12:00:15 GMT
Server: Apache/1.3.0 (Unix)
Last-Modified: Fri, 24 Dic 1999 13:03:32 GMT
Content-Type: image/gif

(datos)...
```

HTTP: GET condicional

- Pasado un tiempo, se vuelve a solicitar el mismo objeto:

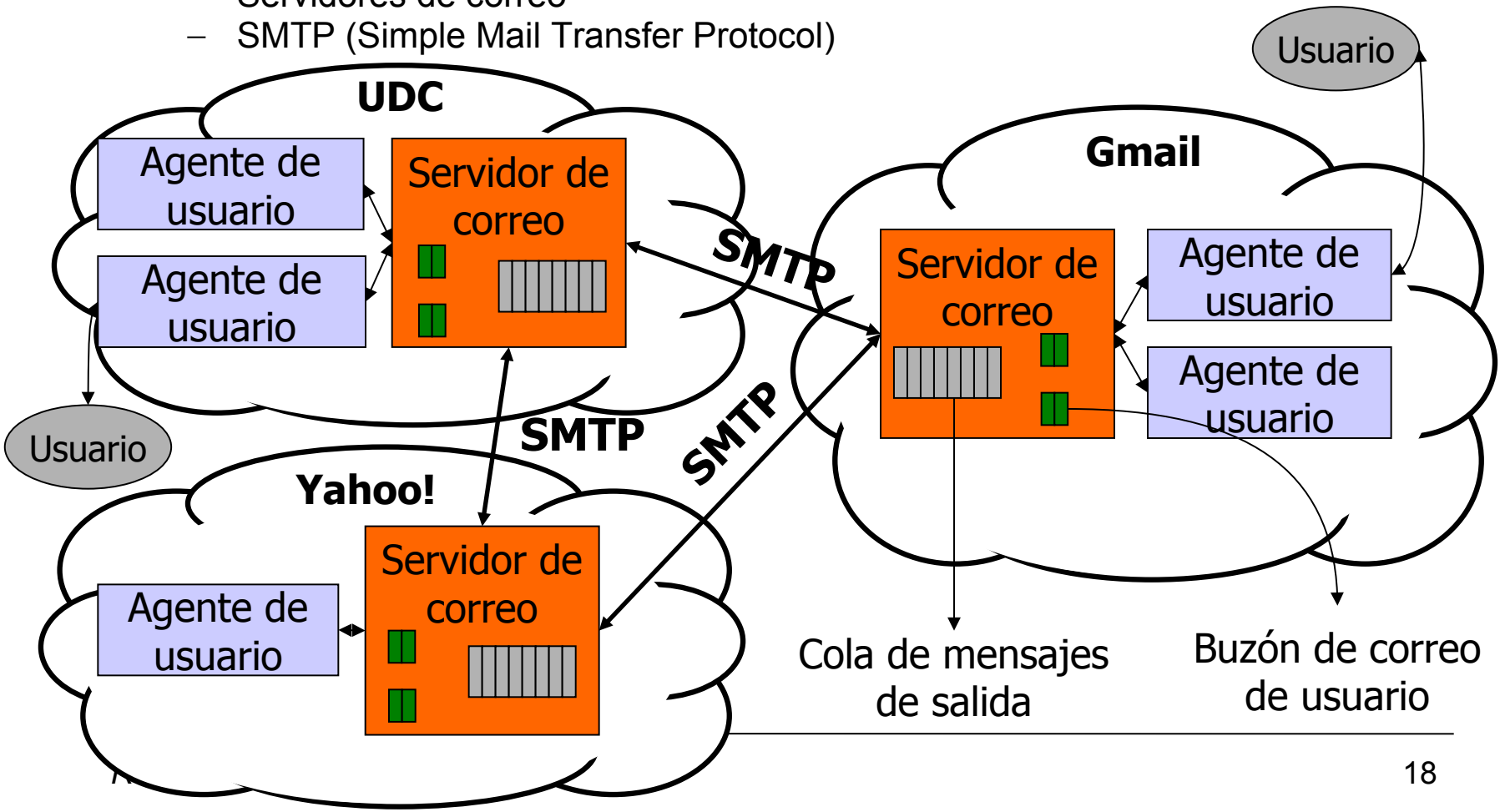
```
GET /images/udc.gif HTTP/1.1
User-agent: Mozilla/4.0
If-modified-since: Fri, 24 Dic 1999 13:03:32 GMT
```

- Si no se ha modificado, el servidor no envía el objeto de nuevo.

```
HTTP/1.1 304 Not Modified
Date: Wed, 5 Jan 2000 20:30:43 GMT
Server: Apache/1.3.0 (Unix)
```

Correo electrónico

- Inventado por Ray Tomlinson en 1971.
- Medio asíncrono de comunicación.
- Componentes:
  - Lectores de correo o agentes de usuario
  - Servidores de correo
  - SMTP (Simple Mail Transfer Protocol)

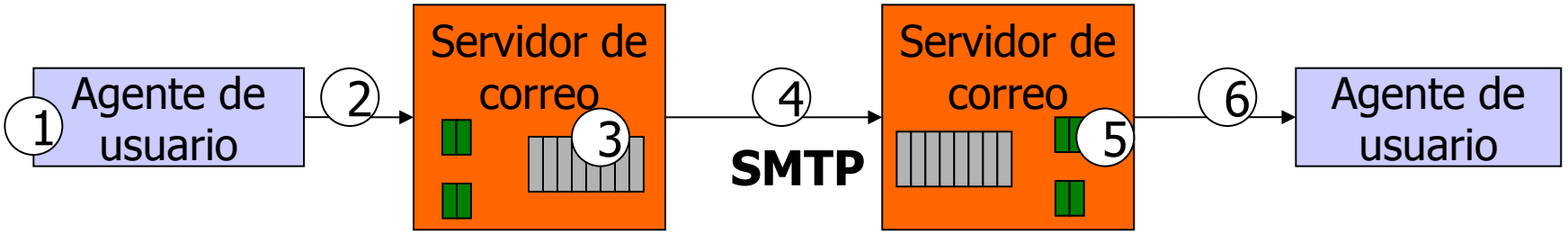


SMTP

- Definido en el RFC 5321.
- Permite el intercambio de mensajes entre servidores de correo.
  - El remitente actúa como cliente
  - El destinatario actúa como servidor
- El cliente SMTP establece una conexión TCP con el puerto 25 del servidor SMTP
  - Si el servidor está fuera de servicio se intentará más tarde.
- Se realiza la sincronización entre emisor y receptor
  - Se indica la dirección de correo electrónico del remitente
- El cliente envía el mensaje
  - Este proceso se repite si hay más mensajes (para el mismo servidor) y después se cierra la conexión TCP.

- SMTP utiliza mensajes en formato ASCII (sólo texto) → Si el mensaje tiene caracteres no ASCII o binarios → Tiene que ser codificado (MIME)
- Es un protocolo de oferta (el cliente envía al servidor), frente a HTTP que es un protocolo de demanda.

SMTP



- Proceso de envío de un mensaje de correo electrónico:
  1. El usuario, mediante su lector de correo, crea el mensaje (p.e. para john.doe@udc.es).
  2. El lector de correo envía el mensaje al servidor de correo del emisor.
     - Se almacena en la cola de mensajes.
  3. El servidor de correo (actuando como cliente SMTP) se conecta al servidor de correo del destinario (mail.udc.es)
  4. El cliente SMTP envía el mensaje.
  5. El servidor SMTP recibe el mensaje y lo almacena en el buzón del destinatario.
  6. El destinatario utiliza su lector de correo para obtener el mensaje.

Formato correo electrónico

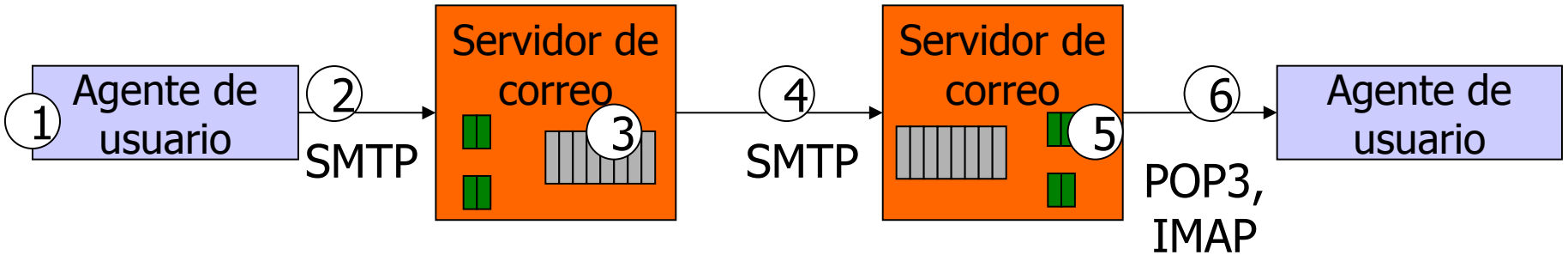
- Un mensaje de correo electrónico consta de dos partes: cabecera y cuerpo (separadas por una línea en blanco).
  - Cabecera: información sobre el correo
  - Cuerpo: el propio correo electrónico
- Algunos campos de la cabecera son:
  - **From**: sólo una por mensaje. Pueden usarse estos formatos:
    - John Doe john.doe@gmail.com
    - john.doe@gmail.com
    - john.doe (John Doe)
  - **To**: una o más por mensaje.
  - **Cc** y **Bcc**
  - **Subject**: tema del mensaje.
  - **Date**: fecha y hora en que el mensaje fue enviado.
  - **Message-Id**: identificador de cada mensaje, insertado por el ordenador remitente.
  - **Received**: información sobre el envío del mensaje, como las máquinas por las que pasó el mensaje.
  - **Reply-To**: dirección a la que se debe responder (no tiene porque coincidir con la del remitente).

MIME

- Multipurpose Internet Mail Extensions
- Definido en varios RFCs: 2045, 2046, 2047, 4288, 4289 y 2049.
- Permite enviar contenidos distintos de texto ASCII en mensajes de correo electrónico. Por ejemplo: idiomas con acentos, no latinos (ruso), sin alfabetos (chino) o contenido binario.
- Sólo afecta a los agentes de usuario, ya que para SMTP es transparente.
- Campos MIME:
  - MIME-Version
  - Content-Description: cadena de texto que describe el contenido. Es necesaria para que el destinatario decida si descodificar y leer el mensaje.
  - Content-Id
  - Content-Transfer-Encoding: indica la manera en que está codificado el cuerpo del mensaje.
  - Content-Type: tipo del cuerpo del mensaje
- Los tipos están cambiando continuamente, pero estos son algunos de los más conocidos y empleados: text/plain, text/richtext, text/html, image/gif, image/jpeg, application/octet-stream, application/postscript, application/msword, audio/basic o video/mpeg

Protocolos de acceso al correo

- ¿Cómo se comunican los lectores de correo con los servidores de correo (pasos 2 y 6)?



- El lector del emisor puede utilizar SMTP
- Pero el lector del receptor no → Se necesita otro protocolo para acceder/leer el correo → POP3, IMAP.

Protocolos de acceso al correo

- POP3 – Post Office Protocolv3: definido en RFC 1939.
- Protocolo de acceso al correo muy simple.
- Modo de operación en tres fases:
  - Autorización: login y password
  - Transacción: recuperar los mensajes, marcar para borrado y estadísticas de correo.
  - Actualización: cuando finaliza la sesión, el servidor de correo borra los mensajes marcados.
- Dos configuraciones del cliente POP3:
  - Descargar y borrar
  - Descargar y guardar

- IMAP – Internet Mail Access Protocol: RFC 3501.
- Permite crear y gestionar buzones remotos (en el servidor de correo).
- IMAP asocia cada mensaje con un buzón.
  - Inicialmente al INBOX
- Proporciona comandos para crear buzones, mover mensajes, buscar mensajes.
- IMAP mantiene información de estado de los usuarios entre sesiones (nombres buzones, mensajes...)
- Dispone de comandos para recuperar componentes de los mensajes.
  - P.e. sólo las cabeceras